Range Quantile Queries: Another Virtue of Wavelet Trees ⋆

Travis Gagie[1], Simon J. Puglisi[2⋆⋆], and Andrew Turpin[2]

[1] Research Group for Combinatorial Algorithms in Bioinformatics,
Bielefeld University, Germany
travis.gagie@gmail.com

[2] School of Computer Science and Information Technology,
Royal Melbourne Institute of Technology, Australia
{simon.puglisi,andrew.turpin}@rmit.edu.au

Abstract. We show how to use a balanced wavelet tree as a data structure that stores a list of numbers and supports efficient *range quantile queries*. A range quantile query takes a rank and the endpoints of a sublist and returns the number with that rank in that sublist. For example, if the rank is half the sublist's length, then the query returns the sublist's median. We also show how these queries can be used to support space-efficient *coloured range reporting* and *document listing*.

1 Introduction

If we are given a list of the closing prices of a stock for the past n days and asked to find the kth lowest price, then we can do so in $\mathcal{O}(n)$ time [2]. We can also preprocess the list in $\mathcal{O}(n \log n)$ time and store it in $\mathcal{O}(n)$ words such that, given k later, we can find the answer in $\mathcal{O}(1)$ time: we simply sort the list. However, we might also later face *range quantile queries*, which have the form "what was the kth lowest price in the interval between the ℓth and the rth days?". Of course, we could precompute the answers to all such queries, but storing them would take $\Omega(n^3 \log n)$ bits of space. In this paper we show how to use a balanced wavelet tree to store the list in $\mathcal{O}(n)$ words such that we can answer range quantile queries in $\mathcal{O}(\log \sigma)$ time, where σ is the number of distinct items in the entire list. We can generalize our result to any constant number of dimensions but, currently, only by using slightly super-linear space.

We know of no previous work on quantile queries[3], but several authors have written about *range median queries*, the special case in which k is half the length of the interval between ℓ and r. Krizanc, Morin and Smid [12] introduced

⋆ This work was supported by the Sofja Kovalevskaja Award from the Alexander von Humboldt Foundation and the German Federal Ministry of Education and Research and by the Australian Research Council.

⋆⋆ Corresponding Author.

[3] Henceforth, for brevity, we will use "quantile query" to mean "range quantile query", and similarly with other types of range queries.

Table 1. Bounds for range median queries.

	space (words)	time	restriction
Krizanc *et al.* [12]	$\mathcal{O}(n)$	$\mathcal{O}(n^\epsilon)$	$\epsilon > 0$
Krizanc *et al.* [12]	$\mathcal{O}(n \log_b n)$	$\mathcal{O}\big(b \log^2 n / \log b\big)$	$2 \leq b \leq n$
Krizanc *et al.* [12]	$\mathcal{O}\big(n \log^2 n / \log\log n\big)$	$\mathcal{O}(\log n)$	
Petersen and Grabowski [17]	$\mathcal{O}\big(n^2 (\log\log n)^2 / \log^2 n\big)$	$\mathcal{O}(1)$	
Theorem 2	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	

the problem of preprocessing for median queries and gave four solutions, three of which have worse bounds than using a balanced wavelet tree; their fourth solution involves storing $\mathcal{O}\big(n^2 \log\log n / \log n\big)$ words to answer queries in $\mathcal{O}(1)$ time. Bose, Kranakis, Morin and Tang [3] then considered approximate queries, and Har-Peled and Muthukrishnan [10] and Gfeller and Sanders [8] considered batched queries. Recently, Krizanc *et al.*'s fourth solution was superseded by one due to Petersen and Grabowski [16, 17], who reduced the space bound to $\mathcal{O}\big(n^2 (\log\log n)^2 / \log^2 n\big)$ words. Table 1 shows the bounds for Krizanc *et al.*'s first three solutions, for Petersen and Grabowski's solution, and for using a balanced wavelet tree.

Har-Peled and Muthukrishnan [10] describe applications of median queries to the analysis of Web advertising logs. In the final section of this paper we show that our solution for quantile queries can be used to support *coloured range reporting*, that is, to enumerate the distinct items in a sublist. This result immediately improves Välimäki and Mäkinen's recent space-efficient solution to the *document listing problem* [14, 19].

In the full version of this paper we will also discuss how to use a wavelet tree to answer range counting queries (see [13]), coloured range counting queries (returning the number of distinct elements in a range without enumerating them), and how to support updates at the cost of slowing queries down to take time proportional to the logarithm of the largest number allowed.

2 Wavelet Trees

Grossi, Gupta and Vitter [9] introduced wavelet trees for use in data compression, and Ferragina, Giancarlo and Manzini [6] showed they have myriad virtues in this respect. Wavelet trees are also important for compressed full-text indexing [15]. As we shall see, there is yet more to this intriguing data structure.

A wavelet tree T for a sequence s of length n is an ordered, strictly binary tree whose leaves are labelled with the distinct elements in s in order from left to right and whose internal nodes store binary strings. The binary string at the root contains n bits and each is set to 0 or 1 depending on whether the corresponding character of s is the label of a leaf in T's left or right subtree. For each internal node v of T, the subtree T_v rooted at v is itself a wavelet tree for the *subsequence* of s consisting of the occurrences of its leaves' labels.

For example, if $s = \mathsf{a}, \mathsf{b}, \mathsf{r}, \mathsf{a}, \mathsf{c}, \mathsf{a}, \mathsf{d}, \mathsf{a}, \mathsf{b}, \mathsf{r}, \mathsf{a}$ and the leaves in T's left subtree are labelled a, b and c, then the root stores 00100010010, the left subtree is a wavelet tree for $\mathsf{abacaaba}$ and the right subtree is a wavelet tree for rdr. The important properties of the wavelet tree for our purposes are summarized in the following lemma.

Theorem 1 (Grossi et al. [9]) *The wavelet tree T for a list of n elements on alphabet σ requires $n \log \sigma(1 + o(1))$ bits of space, and can be constructed in $O(n \log \sigma)$ time.*

To see why the space bound is true, consider that the binary strings' total length is the sum over the distinct elements of their frequencies times their depths, which is $\mathcal{O}(n \log \sigma)$ bits. The construction time bound is easy to see from the recursive description of the wavelet tree given above.

We note as an aside that, while investigating data structures that support rank and select queries, Mäkinen and Navarro [13] pointed out a connection between wavelet trees and a data structure due to Chazelle [4] for two-dimensional range searching on sets of points.

3 Range Quantile Queries

We now describe how the wavelet tree can be used to answer quantile queries. Let s be the list of n numbers we want to query. We build and store the wavelet tree T for s and, at each internal node v, we store a small data structure that lets us perform $\mathcal{O}(1)$-time rank queries on v's binary string. A rank query on a binary string takes a position and returns the number of 1s in the prefix that ends at that position. Jacobson [11] and later Clark [5] showed we can support $\mathcal{O}(1)$-time rank queries on a binary string with a data structure that uses a sublinear number of extra bits, beyond those needed to store the string itself. It follows that the size of this preprocessed wavelet tree remains $\mathcal{O}(n \log \sigma)$ bits.

Given k, ℓ and r and asked to find the kth smallest number in $s[\ell..r]$, we start at the root of T and consider its binary string b. We use the two rank queries $\text{rank}_b(\ell - 1)$ and $\text{rank}_b(r)$ to find the numbers of 0s and 1s in $b[1..\ell - 1]$ and $b[\ell..r]$. If there are more than k copies of 0 in $b[\ell..r]$, then our target is a label on one of the leaves in T's left subtree, so we set ℓ to one more than the number of 0s in $b[1..\ell - 1]$, set r to the number of 0s in $b[1..r]$, and recurse on the left subtree. Otherwise, our target is a label on one of the leaves in T's right subtree, so we subtract from k the number of 0s in $b[\ell..r]$, set ℓ to one more than the number of 1s in $b[1..\ell - 1]$, set r to the number of 1s in $b[1..r]$, and recurse on the right subtree. When we reach a leaf, we return its label. An example is given in Figure 1. Since T is balanced and we spend constant time at each node as we descend (using the rank structures), our search takes $\mathcal{O}(\log \sigma)$ time. Thus, together with Theorem 1 we have the following.

Theorem 2 *There exists a data structure of size $\mathcal{O}(n \log \sigma)$ bits which can be built in $\mathcal{O}(n \log \sigma)$ time that answers range quantile queries on $s[1..n]$ in $\mathcal{O}(\log \sigma)$ time.*

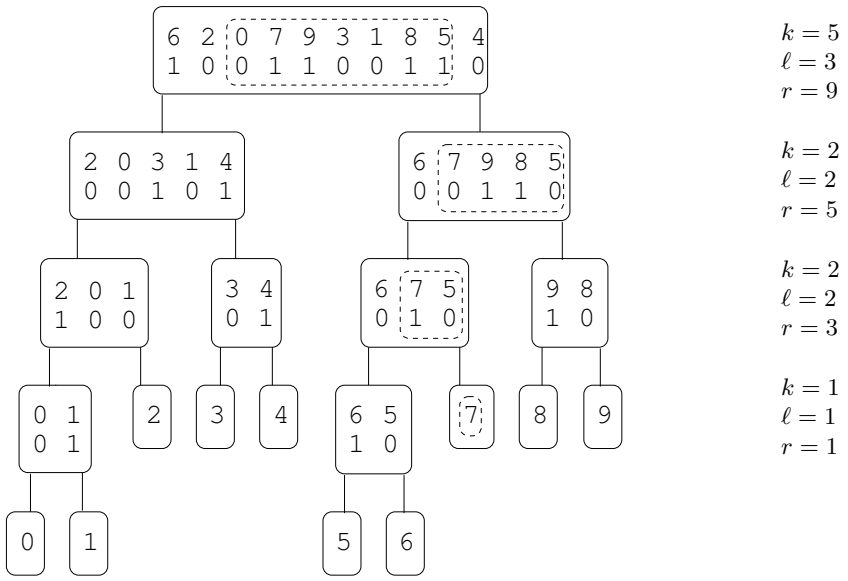

Fig. 1. A wavelet tree T (left) for $s = 6, 2, 0, 7, 9, 3, 1, 8, 5, 4$, and the values (right) the variables k, ℓ and r take on as we search for the 5th smallest element in $s[3..9]$. The dashed boxes in T show the ranges from which we recursively select.

Some comments on σ are in order at this point. Firstly, and obviously, if σ is constant, then so is our query time. If we represent the binary strings at each level of the wavelet tree with a more complicated rank/select data structure of Raman et. al [18] (instead of Clark [5], see [9, 13]), the size of the wavelet tree is reduced to $nH_0(s) + \mathcal{O}(n \log\log n / \log_\sigma n)$ bits without affecting the query time, where $H_0(s)$ is the zeroth order entropy of s. Prior solutions for median queries do not make such *opportunistic* use of space.

At the other extreme, if σ is $\Omega(n)$ we can map the symbols in s to the range $[1..n]$, by first sorting the items in $\mathcal{O}(n \log n)$ time, and storing the mapping in $\mathcal{O}(n \log \sigma)$ bits of space. Preprocessing the array this way, and then using the wavelet tree approach above, allows us to match the $\Omega(n \log n)$ time lower bound for median queries [12], when the number of queries is $\mathcal{O}(n)$. This lower bound applies to any computational model which has an $\Omega(n \log n)$ time lower bound on sorting s. Still, the solution is not completely satisfying, and we leave an open question: Does an $\mathcal{O}(n \log n)$ preprocessing algorithm exist that allows quantile (or even just median) queries to be answered in $o(\log n)$ time when σ is $\Omega(n)$?

It is not difficult to generalize Theorem 2 to any constant number of dimensions, using slightly super-linear space. Suppose we are given a given a multidimensional array A of total size N. We build a balanced binary search tree on the σ' distinct elements in A and, at each node v, we store a binary array of size N with 1s indicating the positions of occurrences of elements in v's subtree. We store each binary array in a folklore data structure (see, e.g., [1, Lemma 2])

that supports multidimensional range counting in $\mathcal{O}(1)$ time using $\mathcal{O}(mN^{\epsilon})$ bits, where m is the number of 1s and ϵ is any positive constant; thus, we use a total of $\mathcal{O}\big(N^{1+\epsilon}\log\sigma'\big)$ bits. To find the kth smallest number in a given range in A, we start at the root of the tree and use a range counting query to find the numbers of 0s and 1s in the same range of the binary array stored there. If there are more than k copies of 0 in the range, then we recurse on the left subtree; otherwise, we subtract the number of 0s from k and recurse on the right subtree. Since we use a single range counting query at each node as we descend, we use a total of $\mathcal{O}(\log\sigma')$ time.

Theorem 1. *For any constants d and $\epsilon > 0$, there exists a data structure of size $\mathcal{O}\big(N^{1+\epsilon}\log\sigma'\big)$ bits that answers d-dimensional range quantile queries on A in $\mathcal{O}(\log\sigma')$ time.*

4 Application to Space Efficient Document Listing

The algorithm for quantile queries just described can, when coupled with another wavelet tree property, be used to enumerate the d distinct items in a given sublist $s[\ell..r]$ in $\mathcal{O}(d\log\sigma)$ time as follows. Let $c_1, c_2, \ldots, c_d$ be the distinct elements in $s[\ell..r]$ and, without loss of generality, assume $c_1 < c_2 < \ldots < c_d$. Further, let m_i, $i \in 1..d$ be the number of times c_i occurs in $s[\ell..r]$. To enumerate the c_i, we begin by finding c_1, which can be achieved in $\mathcal{O}(\log\sigma)$ via a quantile query, as c_1 must be the element with rank 1 in $s[\ell..r]$. Observe now that c_2 must be the element in the range with rank $m_1 + 1$, and in general c_i is the element with rank $1 + \sum_{j=1}^{i-1} m_{j+1}$. Fortunately, each m_i can be determined in $\mathcal{O}(\log\sigma)$ time by exploiting a well known property of wavelet trees, namely, their ability to return, in $\mathcal{O}(\log\sigma)$ the number of occurrences of a symbol in a prefix of s (see [9]). Each m_i is the difference of two such queries.

The *document listing problem* [14] is a variation on the classical pattern matching problem. Instead of returning all the positions at which a pattern P occurs in the text T, we consider T as a collection of k documents (concatenated) and our task is to return the set of documents in which P occurs.

Muthukrishnan [14], who first considered the problem, gave an $\mathcal{O}(n\log n)$ bit data structure (essentially a heavily preprocessed suffix tree) that lists documents in optimal $\mathcal{O}(|P| + ndoc)$ time, where *ndoc* is the number of documents containing P. Recently, Välimäki and Mäkinen [19] used more modern compressed and succinct data structures to reduce the space requirements of Muthukrishnan's approach at the cost of slightly increasing search to $\mathcal{O}(|P| + ndoc\log k)$ time. Their data structure consists of three pieces: the *compressed suffix array* (CSA) of T; a wavelet tree built on an auxilliary array, E (described shortly); and a succinct range minimum query data structure [7].

Central to both Muthukrishnan's and Välimäki and Mäkinen's solutions is the so-called "document array" $E[1..n]$, which is parallel to the suffix array $\mathrm{SA}[1..n]$: $E[i]$ is the document in which suffix $\mathrm{SA}[i]$ begins. Given an interval $\mathrm{SA}[i..j]$ where all the occurrences of a pattern lie, the document listing problem

then reduces to enumerating the distinct items in $E[i..j]$. Without getting into too many details, Välimäki and Mäkinen use the *compressed suffix array* (CSA) of T to find the relevant sublist of E in $\mathcal{O}(|P|)$ time, and then a combination of E's wavelet tree and a range minimum query data structure [7] to enumerate the distinct items in that sublist in $\mathcal{O}(ndoc \log k)$ time. However, as we have described above, the wavelet tree of E alone is sufficient to solve this problem in the same $\mathcal{O}(ndoc \log k)$ time bound. In practice we may expect this new approach to be faster, as the avoidance of the minimum queries should reduce CPU cache misses. Also, because the wavelet tree of E is already present in [19] we have reduced the size of their data structure by $2n + o(n)$ bits, the size of the data structure for minimum queries.

Acknowledgements

Our thanks go to the three anonymous reveiwers whose helpful comments materially improved the paper, and to Meg Gagie for righting our grammar.

References

1. : Space efficient multi-dimensional range reporting. In: *Proceedings of the 15th Conference on Computing and Combinatorics*. (2009) 215–224
2. Blum, M., Floyd, R.W., Pratt, V.R., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. *Journal of Computer and System Sciences* **7** (1973) 448–461
3. Bose, P., Kranakis, E., Morin, P., Tang, Y.: Approximate range mode and range median queries. In: *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science*. (2005) 377–388
4. Chazelle, B.: A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing* **17** (1988) 427–462
5. Clark, D.: *Compact PAT trees*. PhD thesis, Waterloo University, Canada (1996)
6. Ferragina, P., Giancarlo, R., Manzini, G.: The myriad virtues of wavelet trees. In: *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*. (2006) 560–571
7. Fischer, J.: *Efficient Data Structures for String Algorithms*. PhD thesis, LMU, München (2007)
8. Gfeller, B., Sanders, P.: Towards optimal range medians. arXiv:0901.1761 (2009)
9. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: *Proceedings of the 14th Symposium on Discrete Algorithms*. (2003) 841–850
10. Har-Peled, S., Muthukrishnan, S.: Range medians. In: *Proceedings of the 16th European Symposium on Algorithms*. (2008) 503–514
11. Jacobson, G.: Space-efficient static trees and graphs. In: *Proceedings of the 30th Symposium on Foundations of Computer Science*. (1989) 549–554
12. Krizanc, D., Morin, P., Smid, M.H.M.: Range mode and range median queries on lists and trees. *Nordic Journal of Computing* **12** (2005) 1–17
13. Mäkinen, V., Navarro, G.: Rank and select revisited and extended. *Theoretical Computer Science* **387** (2007) 332–347
14. Muthukrishnan, S.: Efficient algorithms for document retrieval problems. In: *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. (2002) 657–666

15. Navarro, G., Mäkinen, V.: Compressed full text indexes. *ACM Computing Surveys* **39** (2007) Article 2
16. Petersen, H.: Improved bounds for range mode and range median queries. In: *Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science*. (2008) 418–423
17. Petersen, H., Grabowski, S.: Range mode and range median queries in constant time and sub-quadratic space. *Information Processing Letters* **109** (2009) 225–228
18. Raman, R., Raman, V., Rao, S.S.: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In: *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. (2002) 233–242
19. Välimäki, N., Mäkinen, V.: Space-efficient algorithms for document retrieval. In: *Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM)*. (2007) 205–215